Algorytm kNN (k – najbliższych sąsiadów)

Wstęp

Założenia – Podobne problemy mają podobne rozwiązania.
Jeśli wiemy jak rozwiązano najbardziej podobny/podobne problemy do naszego problemu to nowy problem należy rozwiązać w sposób identyczny.

Szukając najbardziej podobnego rozwiązania uzyskujemy algorytm 1NN, Czasami jednak dobrze jest poszukać kilku najbardziej podobnych rozwiązań i podjąć działania takie jakie „zwykle" podejmowaliśmy w przeszłości. Oznacza to, że szukamy *k* najbardziej podobnych rozwiązań, następnie liczymy ile raz spośród owych *k* najbardziej podobnych rozwiązań wybraliśmy określone działanie i wybieramy to rozwiązanie które było najpopularniejsze.

Podobieństwo najprościej oceniać poprzez obliczanie odległości. Im mniejsza odległość tym bardziej podobny przypadek

Miary odległości:
Odpowiednio: Euklidesa, Manhattan, Czebyszewa

$$D(\mathbf{x},\mathbf{y})^2 = \sum_{i=1}^{n}(x_i - y_i)^2$$

$$D(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n}|x_i - y_i|$$

$$D(\mathbf{x},\mathbf{y}) = \max_{i=1:n}(|x_i - y_i|)$$

Wstępne przetwarzanie danych

Aby polepszyć działanie algorytmu kNN powszechnie stosowaną techniką jest standaryzacja lub normalizacja danych. Jej zastosowanie powoduje że wszystkie wymiary dla których liczona jest odległość posiadają jednakową istotność. W przeciwnym przypadku mogłoby dojść do sytuacji w której pojedynczy wymiar dominowałby inne wymiary. Np. wzrost wyrażony w cm dominowałby rozmiar stopy również wyrażony w cm. Głównym powodem takiej sytuacji jest to, iż wzrost jest zmienną z przedziału 140cm – 200cm, podczas gdy stopa 20cm – 30cm, w związku z tym dwie osoby o tym samym rozmiarze stopy ale skrajnie różnym wzroście dzieliła by odległość $((200\text{-}140)^2 + (25\text{-}25)^2)^{0.5} = 60$, podczas gdy dla przypadku ludzi o tym samym wzroście ale skrajnie różnym rozmiarze stopy dzieliłaby odległość $((30\text{-}20)^2+(170\text{-}170)^2)^{0.5} = 10$. Widać stąd iż wzrost dominuje odległość, bo znacznie bardziej wpływa na wartość odległości, podczas gdy dwa skrajne przypadki opisane powyżej powinny się cechować tą samą odległością. Powyższy efekt można częściowo usunąć poprzez normalizację lub standaryzację danych.

Standaryzacja

Standaryzacja polega na doprowadzeniu do sytuacji w której wartość średnia poszczególnej cechy ma wartość 0 a odchylenie standardowe = 1

Standaryzacja wyraża się wzorem:

$$x_j(i) = \frac{x_j(i) - mean\left(\mathbf{x}_j\right)}{std\left(\mathbf{x}_j\right)}$$

Gdzie

- i – kolejny indeks wektora
- j – indeks cechy (zmiennej)
- mean(x_j) – wartość średnia zmiennej j
- std(x_j) – odchylenie standardowe zmiennej j

Uwaga
Standaryzując zbiór testowy skorzystaj wartości średniej oraz odchylenia standardowego wyznaczonego na zbiorze treningowym!

Normalizacja

Normalizacja polega na doprowadzeniu do sytuacji w której wartości zmiennej należą do przedziału [0,1]
Normalizacja wyraża się wzorem:

$$x_j(i) = \frac{x_j(i) - \min\left(\mathbf{x}_j\right)}{\max\left(\mathbf{x}_j\right) - \min\left(\mathbf{x}_j\right)}$$

Gdzie

- i – kolejny indeks wektora
- j – indeks cechy (zmiennej)
- max(x_j) – maksymalna wartość zmiennej j
- min(x_j) – minimalna wartość zmiennej j

Uwaga
Normalizując zbiór testowy skorzystaj z wartości max oraz min wyznaczonych na zbiorze treningowym!

Opis algorytmu kNN

Uczenie

1. Dokonaj alternatywnie: standaryzacji/normalizacji/pozostaw dane jakie są
2. Zapamiętaj cały zbiór treningowy

Testowanie

1. Dokonaj standaryzacji/normalizacji/pozostaw dane jakie są (testowanie)
2. Policz odległości pomiędzy wektorem testowym a wszystkimi wektorami zbioru treningowego
3. Posortuj odległości od największej do najmniejszej
4. Zobacz etykiety *k* – najbliższych wektorów do wektora testowego. Zrób histogram częstości poszczególnych etykiet spośród „k-najbliższych” (Policz ile i których etykiet było spośród *k* najbliższych)
5. Przypisz najczęściej występującą etykietę jako etykietę wektora testowego
6. Jeśli wystąpił impas (dwie klasy miały taką samą liczbę głosów) rozwiąż problem losowo

W matlabie

1. Napisz funkcję liczenia odległości, wyznaczającą wartość odległości określonego typu (Euklidesa, Manhattan, Czebyszewa) pomiędzy dwoma wektorami. Wynik zapisz jako osobną funkcję o nazwie *d=odleglosc(x,y,typ)*
   Gdzie
   - x,y – wektory pomiędzy którymi należy policzyć odległość
   - typ – string określający typ funkcji odległości
2. Funkcja ucząca algorytmu kNN o postaci *knn = knn_ucz(tr, typ, k)* powinna jedynie dokonać zapamiętania parametrów *tr, typ, k*
   gdzie
   - knn – nauczony klasyfikator
   - tr – zbiór treningowy
   - k – liczba sąsiadów
   - typ – typ funkcji odległości (jeden z powyższych)
3. W funkcji klasyfikującej *res = kNN_test(knn,te)*
   Gdzie:
   - knn – nauczony klasyfikator
   - te – zbiór testowy
   - res – zbiór testowy wraz ze zbiorem etykiet

   policz odległości pomiędzy danym wektorem testowym a wszystkimi zapamiętanymi wektorami treningowymi

For $i=1:n_{te}$

Dokonaj alokacji pamięci dla zmiennej *dist*

For $j=1:n_{tr}$

Dist(j) = Liczenie odległości

End

Posortuj odległości (zmienną *dist*) od najmniejszej do największej

Do realizacji tego punktu skorzystaj z funkcji *[so,idx] = sort(dist)* która sortuje wartości zapisane w tablicy (wektorze/macierzy *dist*). Wartości posortowane zwracane są w zmiennej *so* oraz zwracany jest również indeks wartości po posortowaniu w zmiennej *idx*. Wartość *idx* ma taki sam rozmiar jak *so* wskazując położenie posortowanej wartości w oryginalnym zbiorze danych, tzn mając dane o postaci *dist=[3.5 4.2 1.9 1.3]* wywołując polecenie *[so,idx]=sort(dist)* uzyskamy *so=[1.3 1.9 3.5 4.2], idx=[4 3 1 2]*. Uzyskane wyniki *idx* oznaczają że wartość najmniejsza (pierwsza po posortowaniu) 1.3 w oryginalnym zbiorze *dist* była na pozycji 4, wartość 1.9 była na pozycji 3, wartość 3.5 była na pozycji 1 zaś ostatnia największa wartość 3.5 w zbiorze *dist* była pierwszą wartością.

Skorzystaj ze zmiennej idx i pobierz etykiety k najbliższych sąsiadów. Możesz skorzystać z zapisu idx(1:k) - zwróci on indeksy najbliższych sąsiadów, czyli nr wektorów które stanowią k najbliższych sąsiadów

Policz częstość występowania poszczególnych klas.

Dla ułatwienia etykiety klas można najpierw zapisać w postaci binarnej tak iż każdej klasie odpowiada jedna kolumna np. dla problemu 3 klasowego mamy

Klasa zapis liczbowy	Klasa 1	Klasa 2	Klasa 3
1	1	0	0
2	0	1	0
3	0	0	1

Wówczas możesz skorzystać z zapisu

Su = sum(Tr_lab(idx(1:k),:))
[ms,Res(i)] = max(su)
Gdzie Tr_lab jest tablicą etykiet dla zbioru treningowego zapisaną w sposób binarny

End

Uwagi

Alokacja pamięci w mat labie:

Matlab umożliwia tworzenie tablic o dynamicznie zmieniającym się rozmiarze jednakże okupione jest to dużym nakładem obliczeniowym, co znacznie spowalnia szybkość działania Matlaba. W celu przyspieszenia działania Matlaba dużo lepszym rozwiązaniem jest wstępne zaalokowanie pamięci dla tablicy. Do alokacji pamięci służą funkcje *ones* lub *zeros*. Wywołanie jednej i drugiej funkcji ma postać: *zeros(row,col)* i *ones(row,col)* gdzie *row*-oznacza ilość wierszy w tablicy, natomiast *col*-ilość kolumn w tablicy. Np. *d=zeros(3,1)* spowoduje utworzenie tablicy *d=[0;0;0]* zaś *d=zeros(1,3)* spowoduje utworzenie *d=[0 0 0]* Pamiętaj że symbol *„;”* oznacza kolejne wiersze. Funkcja *ones* działa w identyczny sposób, z tą różnicą iż utworzona tablica wypełniana jest wartościami 1.

Zadanie:

1. Zaimplementuj algorytm kNN wg powyższej zależności
2. Sprawdź działanie algorytmu dla różnych wartości *k,* dla różnych miar odległości oraz różnych metod wstępnego przetwarzania (standaryzacji/normalizacji)
3. Porównania dokonaj na przykładzie zbiorów danych: Iris, WBC, Ionosphere,
4. Zaproponuj i sprawdź najlepszą kombinację parametrów dla zbioru Spam